

OWL or SHACL?

A Beginner's Guide to Making the Right Choice

Connected Data London 2024
December 11, 2024

Tara Raafat, Head of Metadata Strategy, Office of the CTO
Davide D'Amico, Ontology Specialist

TechAtBloomberg.com

© 2024 Bloomberg Finance L.P. All rights reserved.

Bloomberg



TechAtBloomberg.com

Disclaimer: Created by ChatGPT

© 2024 Bloomberg Finance L.P. All rights reserved.

Bloomberg

Engineering

What we are learning today

- This will be an “Example-Based” hands-on tutorial
- Main Focus: Comparing the strengths and weaknesses of OWL and SHACL
- Some basics of OWL and SHACL will be explained through multiple examples
- Together, we will make a choice of OWL or SHACL (or both) to solve problems
- We will understand the implication of the choices we make
- This will **NOT** be an in-depth tutorial of everything OWL or everything SHACL... if only we had more time :)

Some Logistics



Download Protégé Desktop (NOT WebProtégé): <https://protege.stanford.edu/>



Open SHACL playground: <https://shacl-playground.zazuko.com/>



Download ZIP / Clone GitHub repo: https://github.com/SemanticMasterclass/owl_shacl



Models: contains all models for our session



Queries: contains all SPARQL queries for our session



Shapes: contains all SHACL shapes for our session

Bloomberg

Engineering

When you see these icons



Go to Protégé

For modeling in owl
Adding data instances
Using a reasoner
Running SPARQL



Go to SHACL playground

For writing SHACL shapes
For validating data graphs against SHACL shapes



Follow the instructions



A Little Background

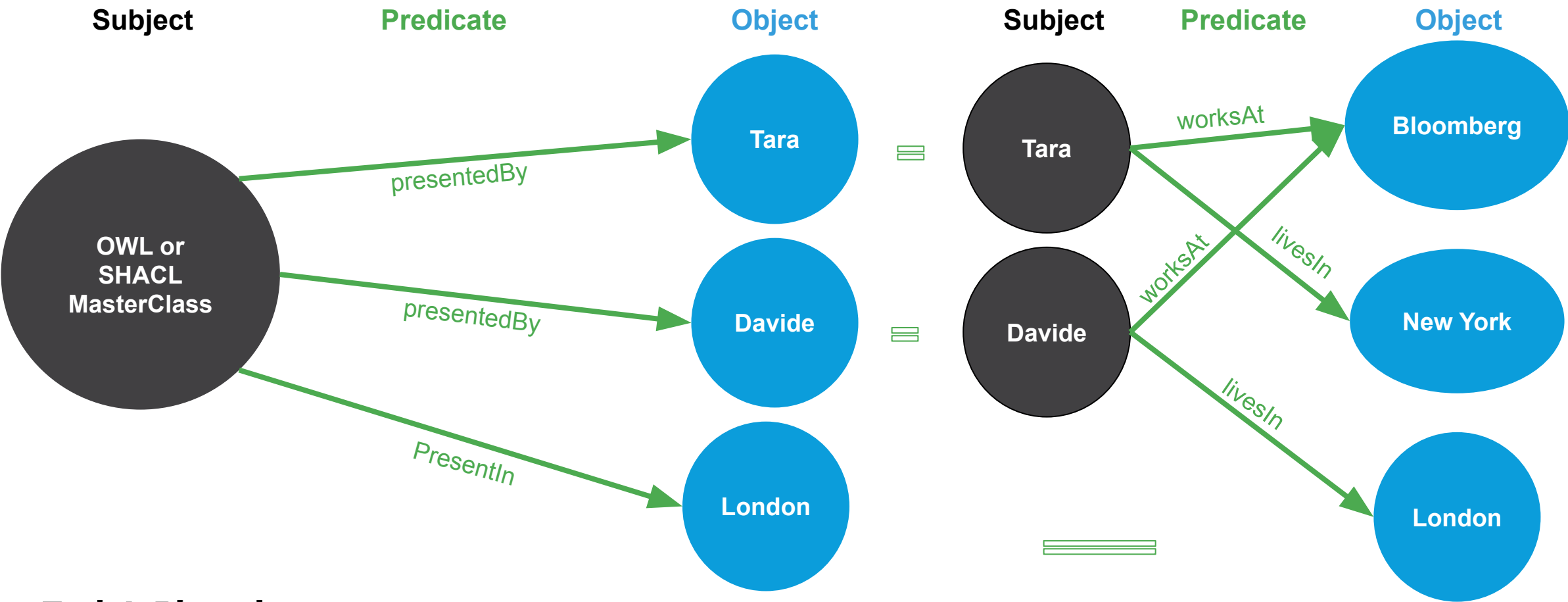
TechAtBloomberg.com

© 2024 Bloomberg Finance L.P. All rights reserved.

Bloomberg

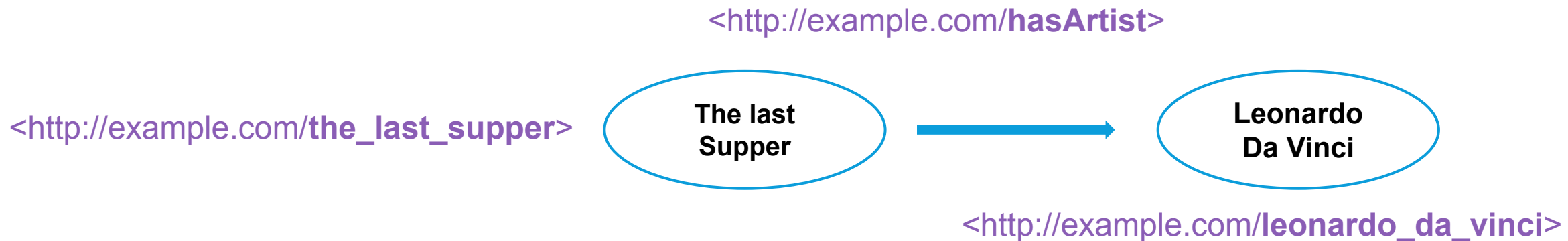
The World in Triples!

(Things not Strings!)



RDF (Resource Description Framework)

- W3C de facto standard for data interchange that is used for representing highly interconnected data
- Simple triple-based model: **<subject> <predicate> <object>**



- Everything in RDF is a resource
- Every resource has a unique identifier
- No schema

Bloomberg

Engineering

A day at the museum....



TechAtBloomberg.com

© 2024 Bloomberg Finance L.P. All rights reserved.

Bloomberg

Engineering

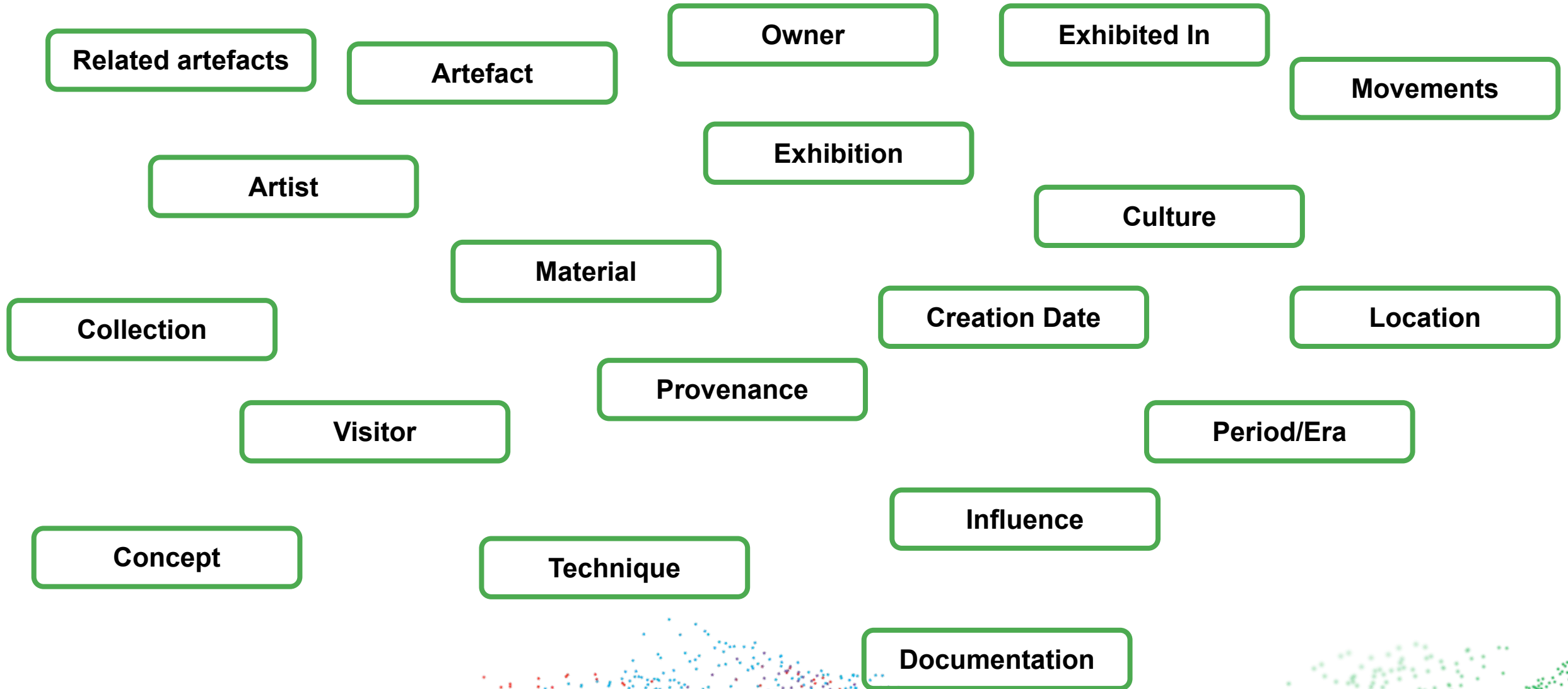
Walking through an exhibitionA million questions !

- What artefacts do you have?
- All the information about them in terms of artist, date of creation, materials, etc.?
- When you create the exhibitions how do you decide what artefacts go together?
- Can you know which city or county an artefact is being exhibited?
- Based on the art movement era of the piece can you tell the artists of those eras?
- Is it possible for you to know whether a copy of an original art pieces you have exists somewhere else?
- Which piece was influenced by whom?

Bloomberg

Engineering

How Do They Organize Their Data?



They use **Ontology Models!!**

TechAtBloomberg.com

© 2024 Bloomberg Finance L.P. All rights reserved.

Bloomberg

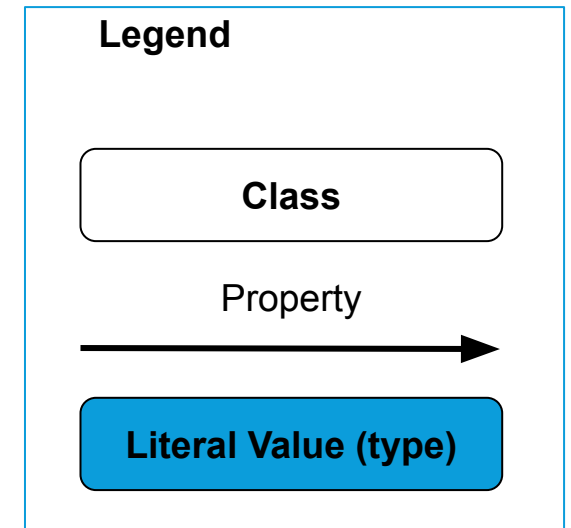
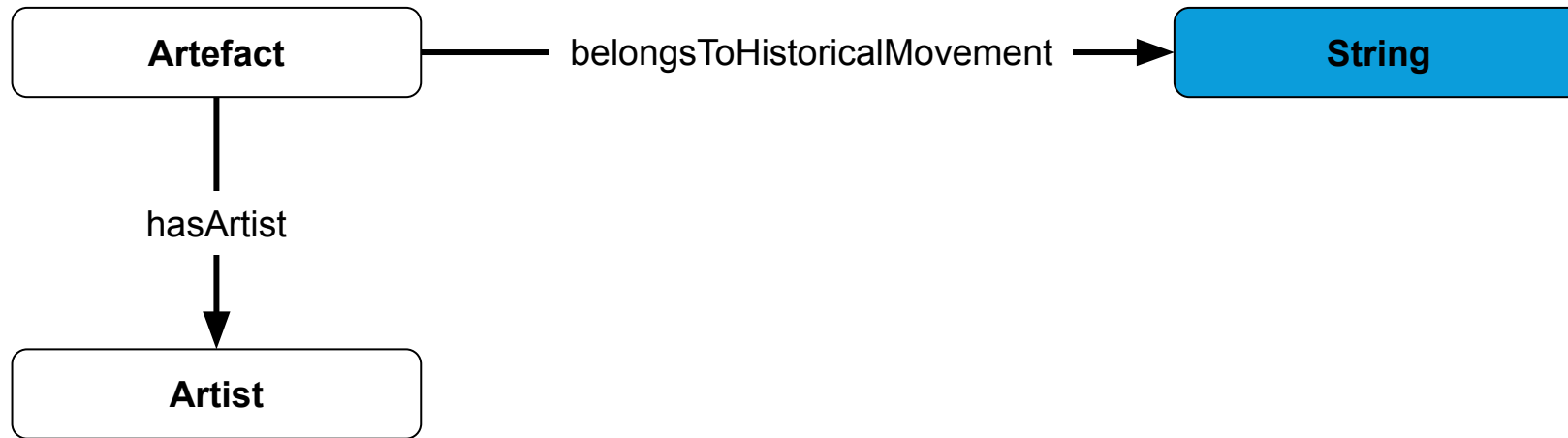
Engineering

How do you decide on the design of the ontology model?

Start with the **competency questions**...

1. Which **artefacts** were created by a certain **artist**?
2. What are the pieces that were **created** in a specific **historical movement**?
3. What **artefacts** were created by a certain **artist** in a certain **historical movement**?

The Simple Ontology



Let's code that in OWL



or

- Open **model_00.ttl**
- Add the above classes and properties
- Open **model_01.ttl**

TechAtBloomberg.com

© 2024 Bloomberg Finance L.P. All rights reserved.

Bloomberg

OWL (Web Ontology Language)

- RDF-based
- A de facto standard for ontology development
- Main components include
 - Classes: define concepts in a domain
 - Properties:
 - Object properties: define relationships between concepts
 - Datatype properties: define relationships between a concept and a literal
 - Individuals: instances of classes
 - Restrictions: Allow definition of cardinality restrictions, as well as existential & universal quantifications (some rules can be defined in the context of a restriction)



Let's add the data



Or

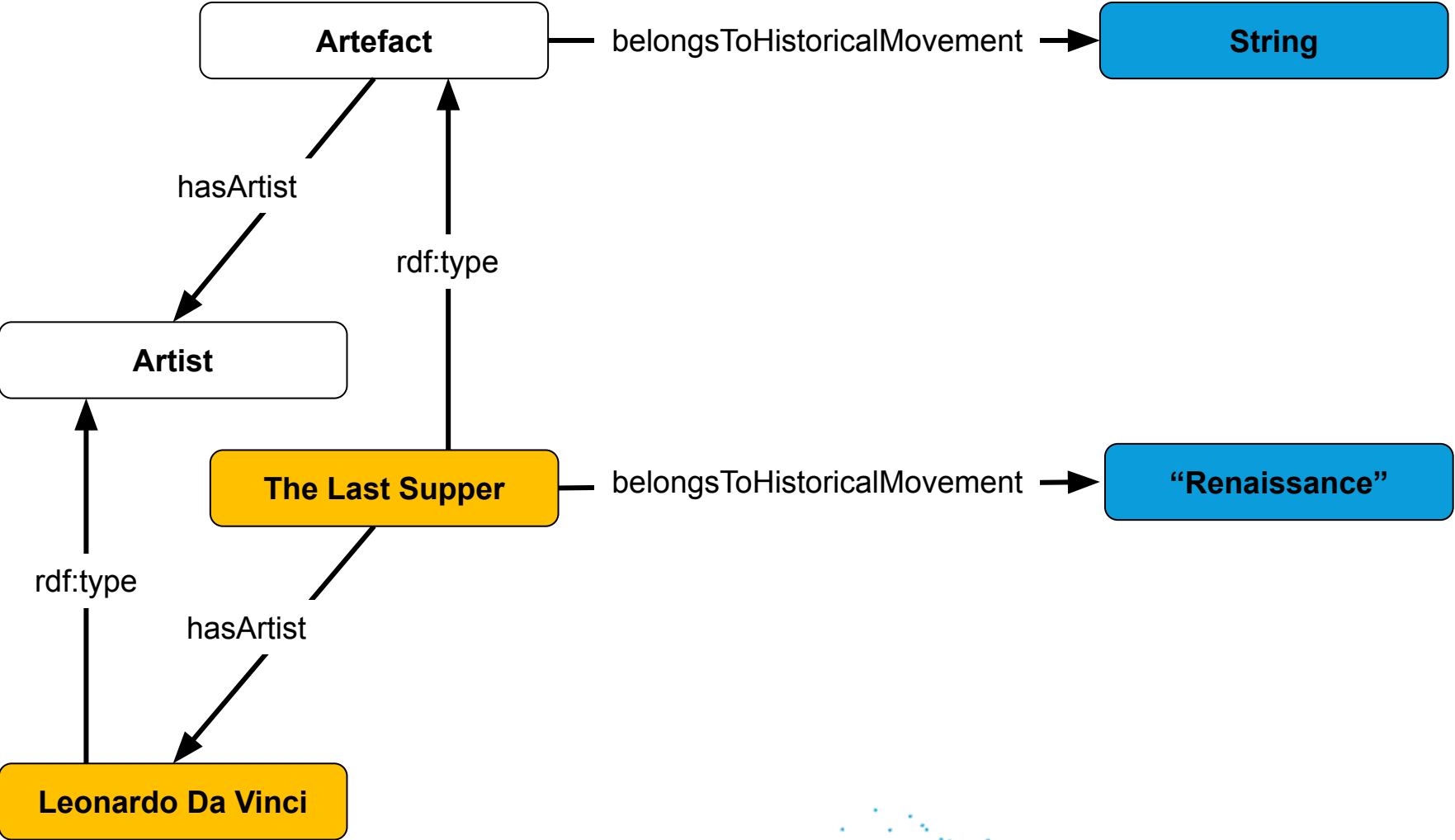
- Create instances and property values
- Open **model_02.ttl**

TechAtBloomberg.com

© 2024 Bloomberg Finance L.P. All rights reserved.

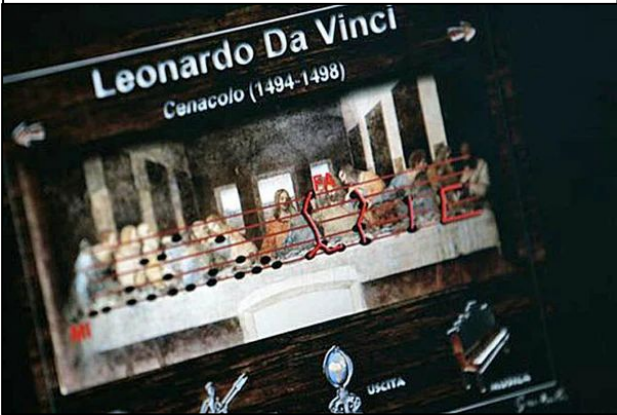
Bloomberg

The Knowledge Graph



Fun Fact:

In 2007, an Italian musician claimed to have found musical notes in *The Last Supper* that result in a tuneful 40-second musical composition. The notes, which are 'hidden' in the bread rolls and hands of the apostles in the painting, is read from right to left, following Da Vinci's own writing style.



Let's test the queries

1. Which artefacts were created by Leonardo da Vinci?
2. What are the pieces that were created during the renaissance movement?
3. What artefacts were created by Leonardo Da Vinci during the Renaissance movement?

```
SELECT ?x
WHERE {
  ?x a ex:Artefact ;
  ex:hasArtist ex:leonardo_da_vinci . }
```

```
SELECT ?x
WHERE {
  ?x a ex:Artefact ;
  ex:belongsToHistoricalMovement
  "Renaissance"^^xsd:string . }
```

```
SELECT ?x
WHERE {
  ?x a ex:Artefact ;
  ex:hasArtist ex:leonardo_da_vinci ;
  ex:belongsToHistoricalMovement
  "Renaissance"^^xsd:string . }
```



Open **model_02.ttl**
Open **Queries.txt**
Run the queries in **Section 1**

Could I have done that with SHACL?

Yes.

[Shape 1](#)



```
ex:ArtefactShape a sh:NodeShape ;  
  sh:targetClass ex:Artefact ;  
  
  sh:property [  
    sh:path ex:belongsToHistoricalMovement ;  
    sh:datatype xsd:string ;  
  ] ;  
  
  sh:property [  
    sh:path ex:hasArtist ;  
    sh:class ex:Artist ;  
  ] .
```

If all you want to do is to answer those three questions, you can use either OWL or SHACL, as they will both give you the basics...

But what if you want to do more?



Let's add more data

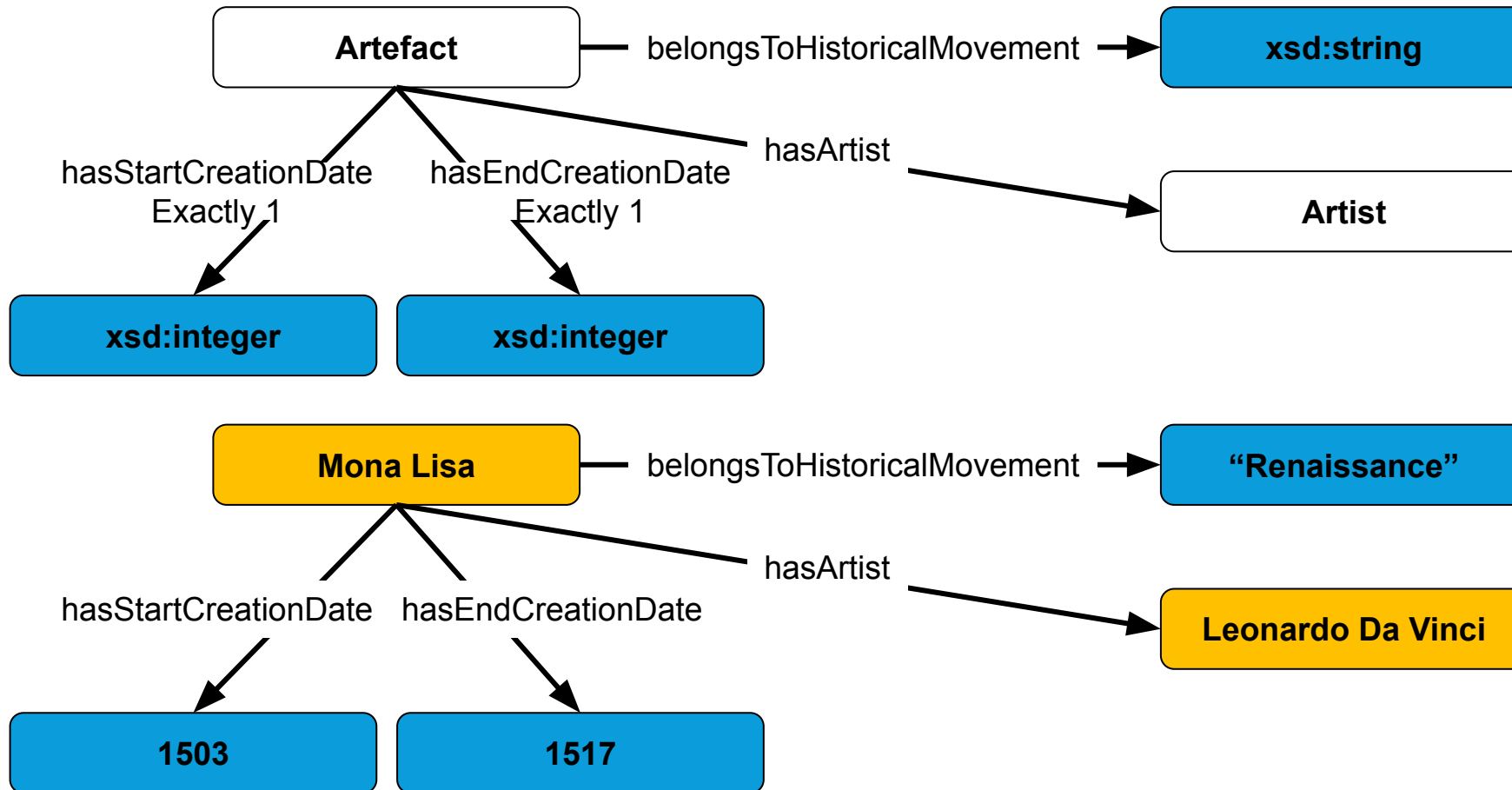


TechAtBloomberg.com

© 2024 Bloomberg Finance L.P. All rights reserved.

Bloomberg

The Knowledge Graph extended



Open **model_03.ttl**

Fun Fact:

Leonardo da Vinci's *Mona Lisa* took 14 years for him to complete, starting in 1503 and continuing until nearly his death in 1517; reportedly, he continuously refined the painting throughout this period.

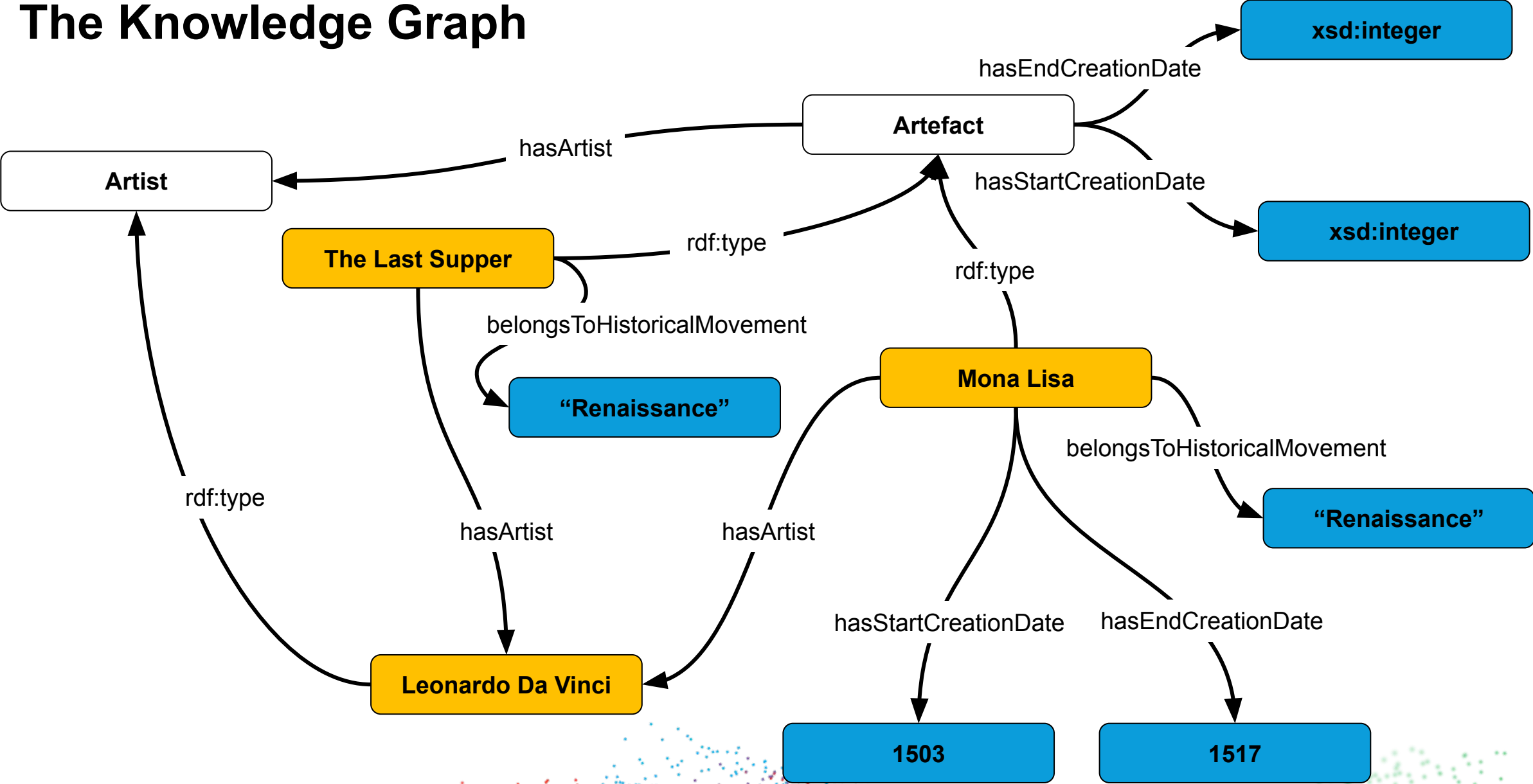


What about **Missing (Incomplete) information**?

Some art pieces do not have full information about them at the time of acquisition.

- How do museum systems handle the missing information?
- Does it cause an error?
- Should it cause an error?

The Knowledge Graph



Would OWL restrictions enforce data's existence & cause an error?

Artefact := hasStartCreationdate **exactly 1** xsd:integer

Artefact := hasEndCreationdate **exactly 1** xsd:integer



Load **model_03.ttl**

Run reasoner

The reasoner **doesn't** throw an error.



Why? Because OWL operates on an Open World Assumption (OWA) and assumes that the absence of information does not imply its absence in reality.





Open World Assumption (OWA)

- If the information doesn't exist, you can't assume it is false. It's just incomplete.
- The reasoner does not return an error.
- When is this useful?
 - In scenarios where not all the knowledge/information is known or available, the OWA and the ontology will support ongoing discovery by remaining **adaptable** and **collaborative**.
- You can update the information without the need to invalidate previous assertions.





**What if you wanted to ensure the data
existed at the time of acquisition?
(data integrity)**



But, what if I want **data integrity**? (ensuring data is there)

- You need to have a **Closed World Assumption (CWA)**, meaning:
 - We need to define the **exact shape** we want our data to be in – exact properties and their restrictions
- We need **SHACL** Shapes
 - Restrictions defined in SHACL shapes have to be met or else there will be an error
 - If something is missing, then **it is false** and we have an error
- Let's see how we ensure that the date of creation is there?

Shape 2

```
ex:ArtefactShape a sh:NodeShape ;
sh:targetClass ex:Artefact ;

sh:property [
  sh:path ex:belongsToHistoricalMovement ;
  sh:datatype xsd:string ;
] ;

sh:property [
  sh:path ex:hasArtist ;
  sh:class ex:Artist ;
] ;

sh:property [
  sh:path ex:hasEndCreationDate ;
  sh:datatype xsd:integer ;
  sh:minCount 1 ;
  sh:maxCount 1 ;
] ;

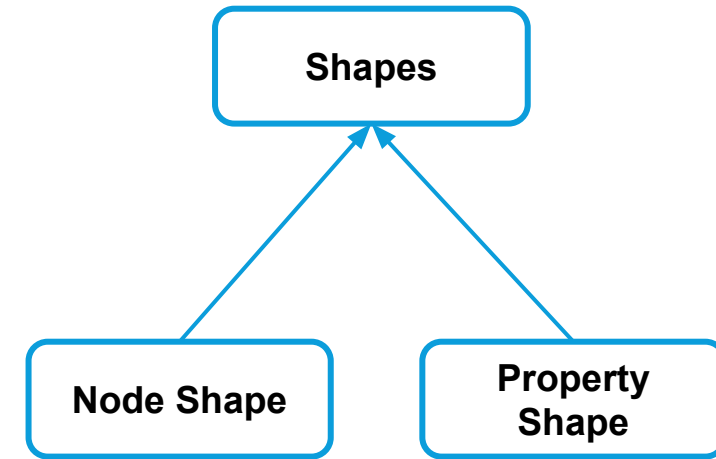
sh:property [
  sh:path ex:hasStartCreationDate ;
  sh:datatype xsd:integer ;
  sh:minCount 1 ;
  sh:maxCount 1 ;
] .
```


The SHACL Shape

A collection of targets and constraints:

- **Targets**: define which nodes in the data graph must conform to the Shape
- **Constraint**: define how to validate a node

declare constraints directly on a node



*declare constraints on the property that is connected to the node through a **path***

The sequence of edges through which a property is connected to a node

Targets

sh:targetClass

```
ex:ArtefactShape a sh:NodeShape ;
  sh:targetClass ex:Artefact ;

  sh:property [
    sh:path ex:belongsToHistoricalMovement ;
  ] ;
```

sh:targetNode

```
ex:Artist a sh:NodeShape .
ex:ArtistShape a sh:NodeShape ;
sh:targetNode ex:Leonardo_Davinci;
sh:nodeKind sh:IRI;
sh:property [sh:path ex:hasDateOfBirth ;
             sh:range xsd:string ].
```

sh:targetSubjectOf

```
ex:SculptureShape a sh:NodeShape ;
sh:targetSubjectOf ex:hasArtist ;
sh:property [
  sh:path ex:hasEndCreationDate ;
  sh:datatype xsd:integer ;
  sh:minCount 1 ;
  sh:maxCount 1 ;
] ;
```

sh:targetObjectOf

```
ex:SculptureShape a sh:NodeShape ;
sh:ObjectOf ex:hasArtist ;
sh:property [
  sh:path ex:hasName ;
  sh:datatype xsd:string ;
  sh:minCount 1 ;
  sh:maxCount 1 ; ] ;
```

Recap so far...

- **Open world:** missing information does not cause an error **OWL**
- **Closed world:** missing information causes an error **SHACL**
 - **Data Integrity:** **SHACL**
 - **Data Validation:** **SHACL**

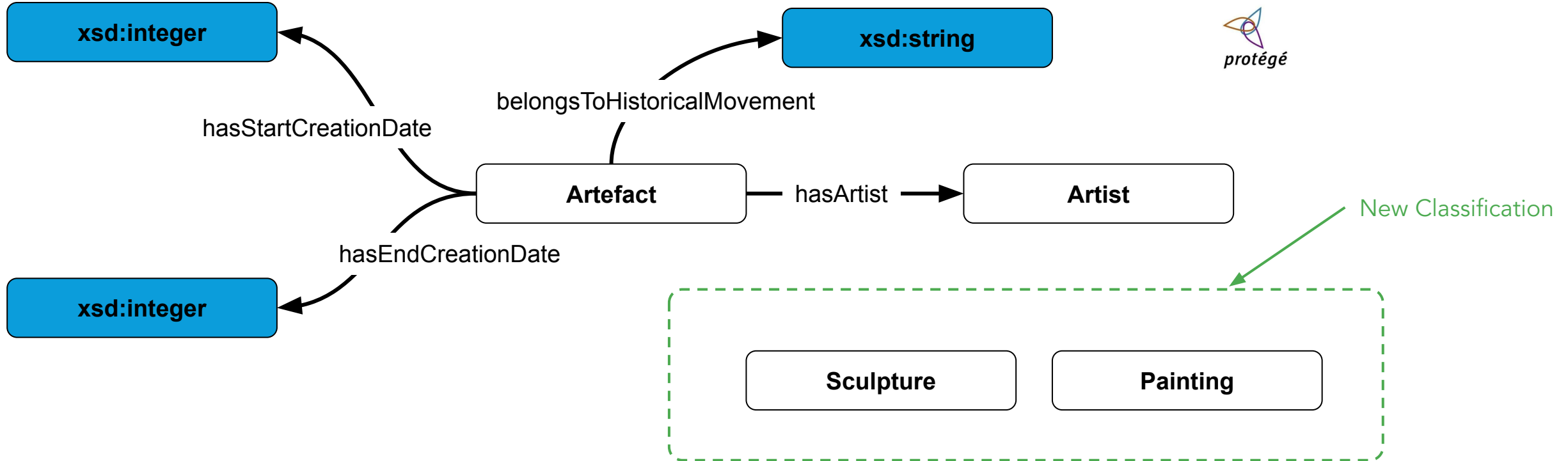
What about **Classification**?

Museums frequently acquire new pieces and classify them according to their own systems. Is there a way to automate those classifications?

Let's say the museum wants to automatically identify whether an artefact is a sculpture or painting at the time of acquisition.

```
ex:david
  ex:belongsToHistoricalMovement "Renaissance" ;
  ex:hasArtist ex:michelangelo ;
  ex:hasEndCreationDate 1504 ;
  ex:hasStartCreationDate 1501 ;
  ex:hasSculptureMaterial "Marble" ;
  rdfs:label "David" .
```


Ontology extension...



Add two new classes Sculpture and Painting
(model_05.ttl)
Run the reasoner

Classification Query

Using the new model and the data we had about David sculpture;
Can we query the data to see what's sculptures we have?

Let's query the data for Sculpture (SPARQL query 2.1):

```
SELECT ?x  
  
WHERE {  
  ?x a ex:Sculpture .  
}
```



Nothing!

Why? There is neither a definition for the class sculpture nor a definition of the property hasSculptureMaterial

Fun Facts:

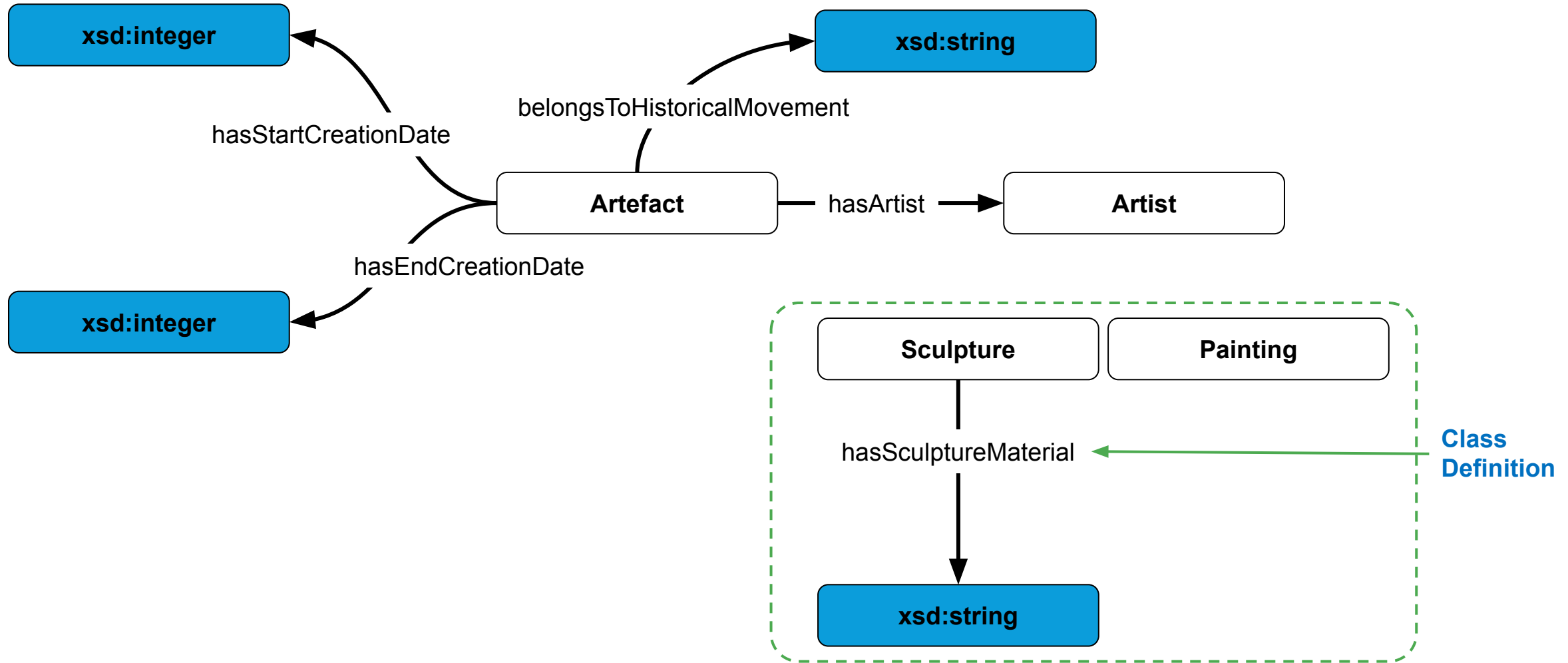
The statue was carved from a single block of marble.

David's right hand is disproportionately large compared to his body because in the Middle Ages, David was said to be "manu fortis", meaning strong of hand.

There are at least 30 full-size replicas of this statue around the world.



Class Definition





Classification by Class Definition



- Add the class definition
OR load **model_06.ttl**
- Run the reasoner
- Add the changes suggested by the reasoner
OR upload **model_06_inferred.ttl**
- Run the query 2.1 again

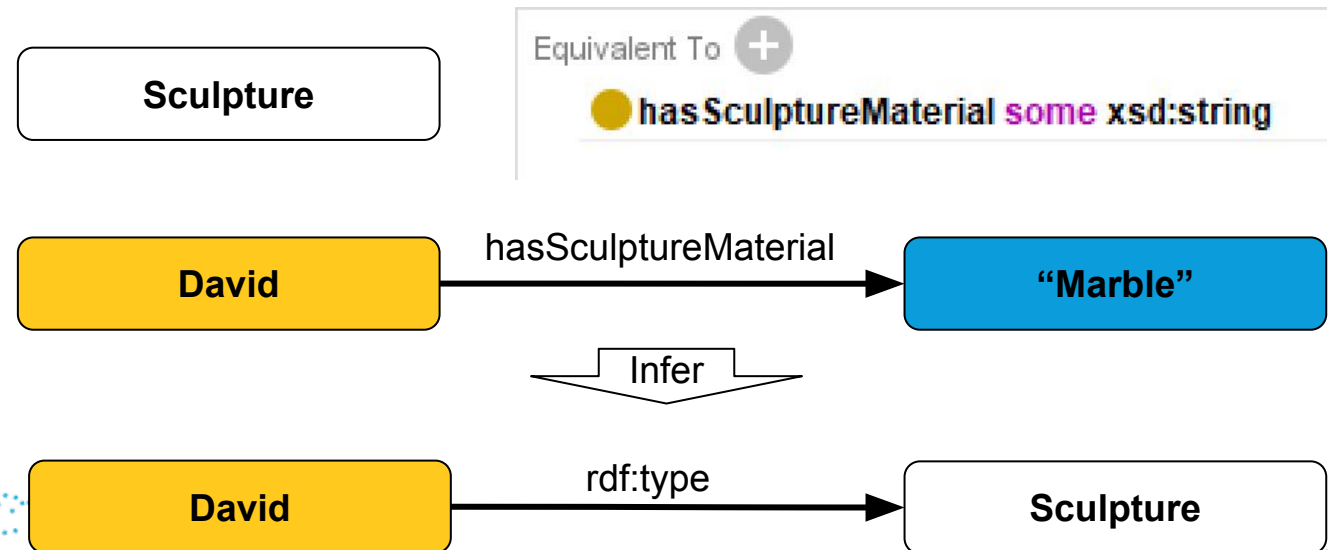
```
SELECT ?x  
  
WHERE {  
  ?x a ex:Sculpture .  
}
```

Class Definition

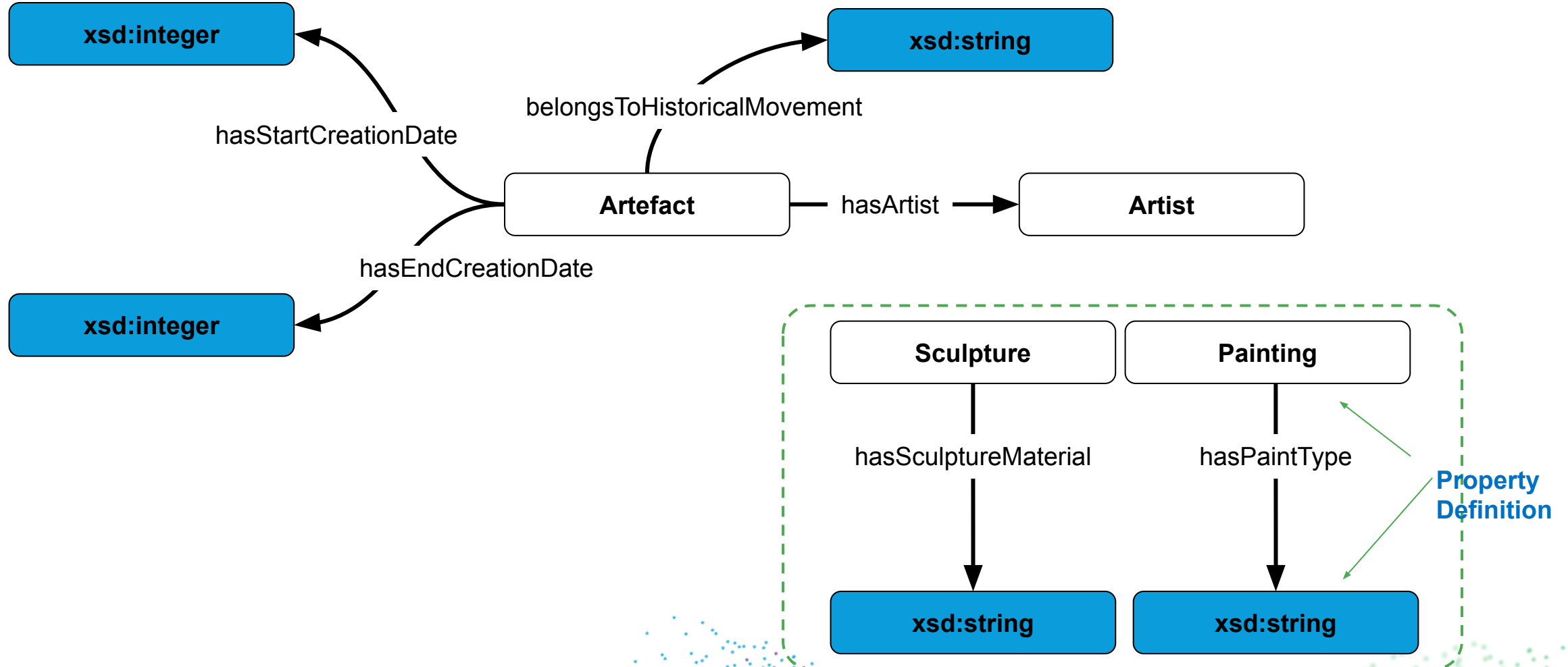
- A **sculpture** has **some** sculpture material



David
Why?



Property Definition





Classification by Property definition



- Add the property definition
OR load **model_07.ttl**
- Run the reasoner
- Add the changes suggested by the reasoner
OR upload **model_07_inferred.ttl**
- Run query 2.2

Property definition:

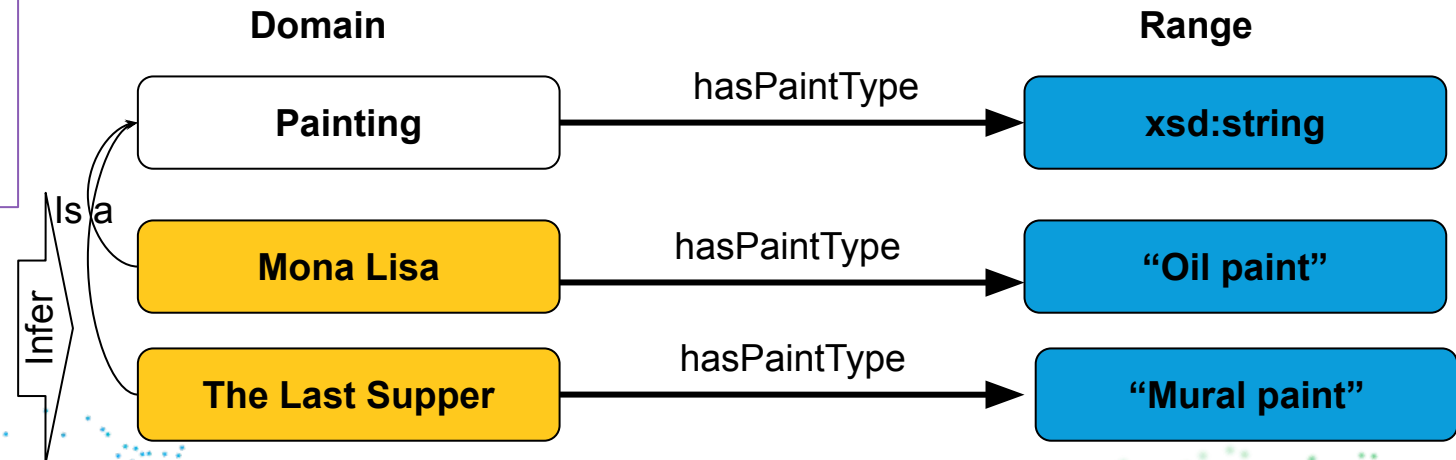
- hasPaintType owl:domain **Painting**
- hasPaintType owl:range xsd:string



Mona Lisa & Last Supper

Why: Based on the “hasPaintType” definition

```
SELECT ?x
WHERE {
  ?x a ex:Painting .
}
```



Classification Query

With all the data we have, can we query to see **what are all the artefacts we have?**

Artefact Query (no. 2.3)

```
SELECT ?x  
  
WHERE {  
  ?x a ex:Artefact .  
}
```

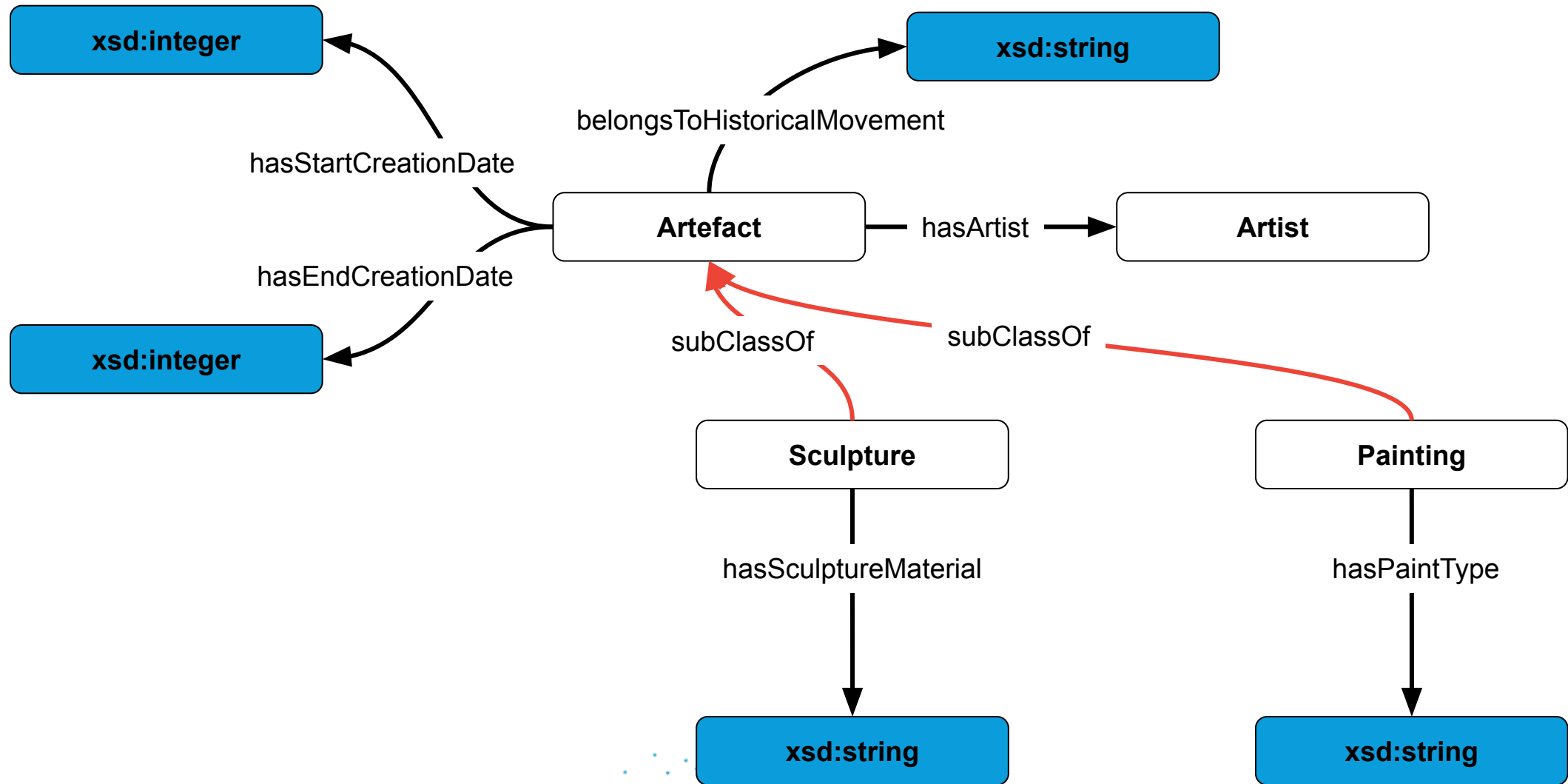
x
ex:mona_lisa
ex:the_last_supper



We did not get “David”!

Why: There’s no subclass relationship defined between the sculpture class and the Artefact class.

Extend the Model





The Subclass inference



- Add the subclasses OR load **model_08.ttl**
- Run sparql 2.4
- Run the reasoner
- Export inferred triples into a new file **OR** load **model_08_inferred.ttl**
- Run the query 2.5

Query 2.4

```
SELECT DISTINCT ?x
WHERE {
  ?x rdf:type/rdfs:subClassOf*
  ex:Artefact .
}
```

Inference/reasoning logic in Query (inference on read)

Query 2.5

```
SELECT DISTINCT ?x
WHERE {
  ?x rdf:type ex:Artefact .
}
```

Inference/reasoning logic applied prior to querying (inference on write)



x
ex:mona_lisa
ex:the_last_supper
ex:david





Inference for Classification

- One of the main reasons to use OWL is its capability of making inferences
- Inference can happen based on:
 - Definition of the class through its properties and restrictions
 - Specifying the domain and range of a property
 - Through the subclass/superclass relationship

On its own, the W3C standard SHACL-Core **does not** support inference

NOTE: For inference, please see SHACL Advance Features (W3C draft document)

Bloomberg

Engineering



So, how do **Restrictions** in OWL & SHACL differ?

OWL & SHACL Restrictions Compared

- In **OWL**, the restriction purpose is **Classification** of resources based on the values they have:
 - If something has these properties and restriction, then it is of this class type, as we saw in the Sculpture example
 - OWL is **property-oriented**, meaning that the definition of a class is based on its properties and the restrictions on those properties
- In **SHACL**, the restriction purpose is for **Validation**, and the logic is reverse meaning:
 - If something is of a class/target type, then it has these properties
 - And, if it doesn't have those properties, it will fail the **validation**

Recap so far...

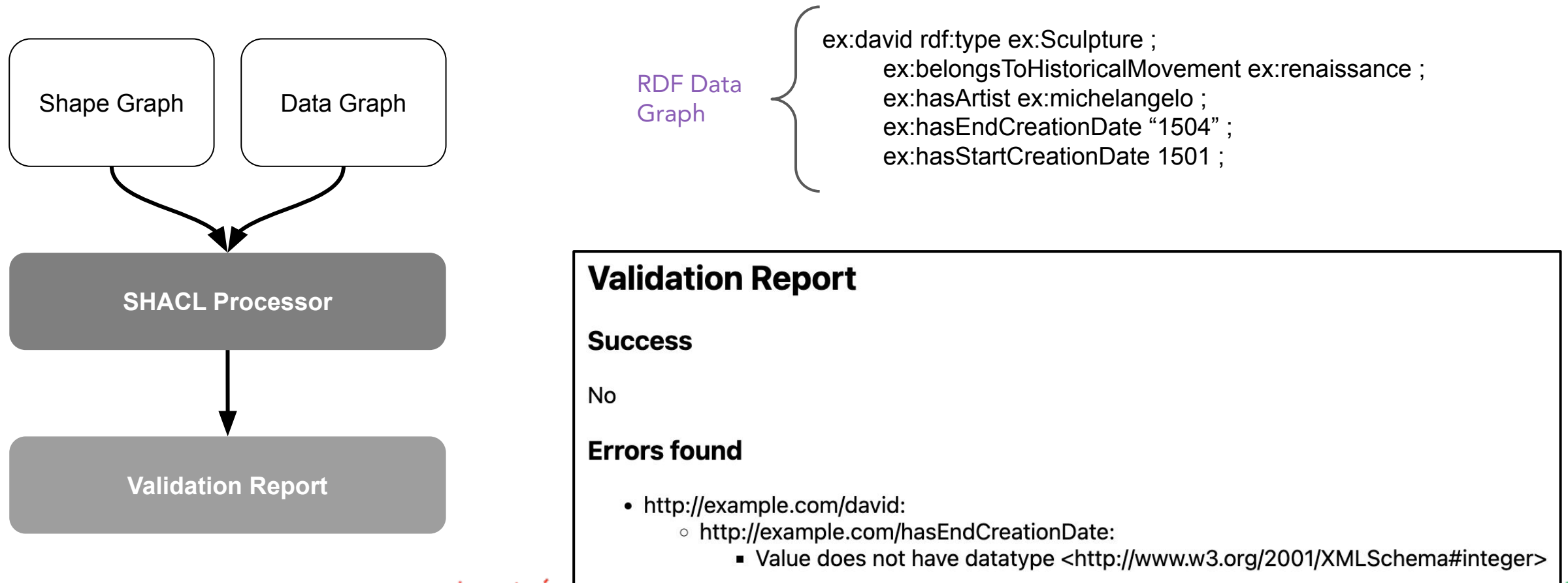
- **Open world:** missing information does not cause an error **OWL**
- **Closed world:** missing information causes an error **SHACL**
 - **Data Integrity : SHACL**
 - **Data Validation : SHACL**
- **Inference for Classification: OWL**
 - Class definition
 - Property domain and range
 - Subclass relationship



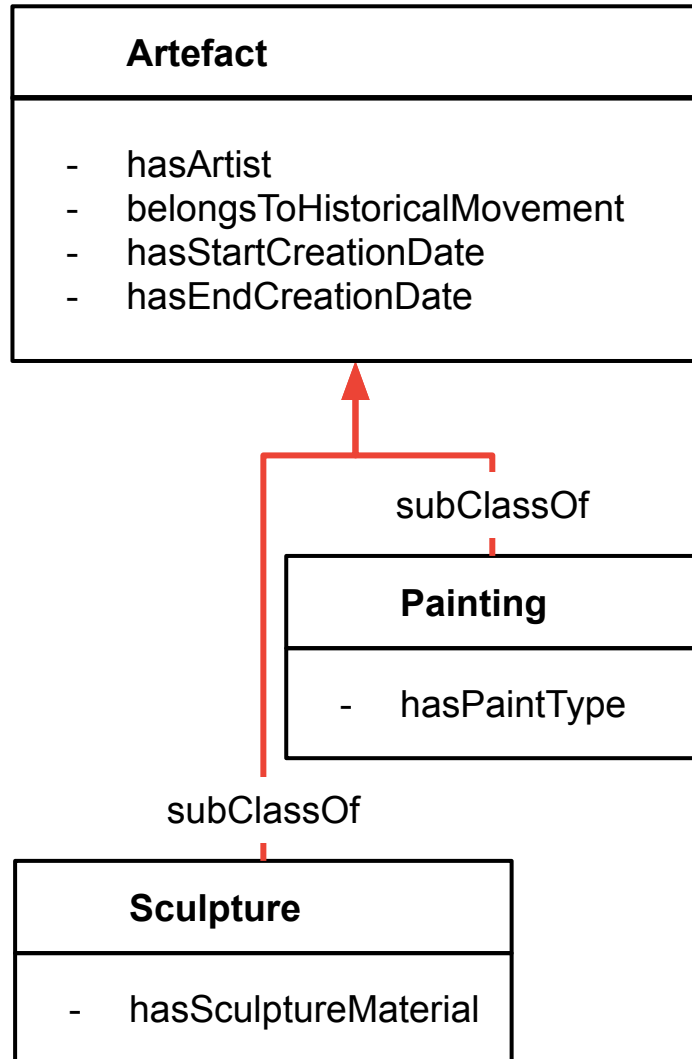
Another validation example

Validating with SHACL

Let's test our shape with the data in SHACL Playground



Inheritance



- Run [Shape 3](#) -> data graph conforms to the Shape so no error
- Change the date of creation from integer to string for one of the paintings



The expectation is to inherit all properties of the super class Artefact -> but that didn't happen!

WHY? Defining the subclass relationship as part of the SHACL shape itself does not enforce inheritance.



But, there is a solution:

- Move the subclass definition to the datagraph [Shape 5](#)
- Or create node relationship between shapes.
- Now, test the datagraph

Error!



WHY? The SHACL playground expects the definition of subclass relationships to be present in the data graph. Other toolings which can load the the shape graph or the ontology with the data will not have this problem



- To have some more fun: try closing the shapes [Shape 6](#)



What if data takes multiple shapes?

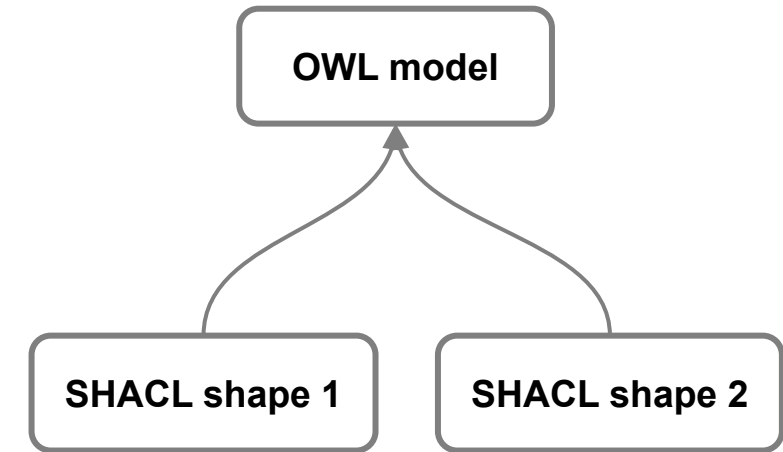
One model, multiple shapes



What if another museum has the same data, but they also require the need to ensure capturing the current owner of the piece or the seller at the time of acquisition?

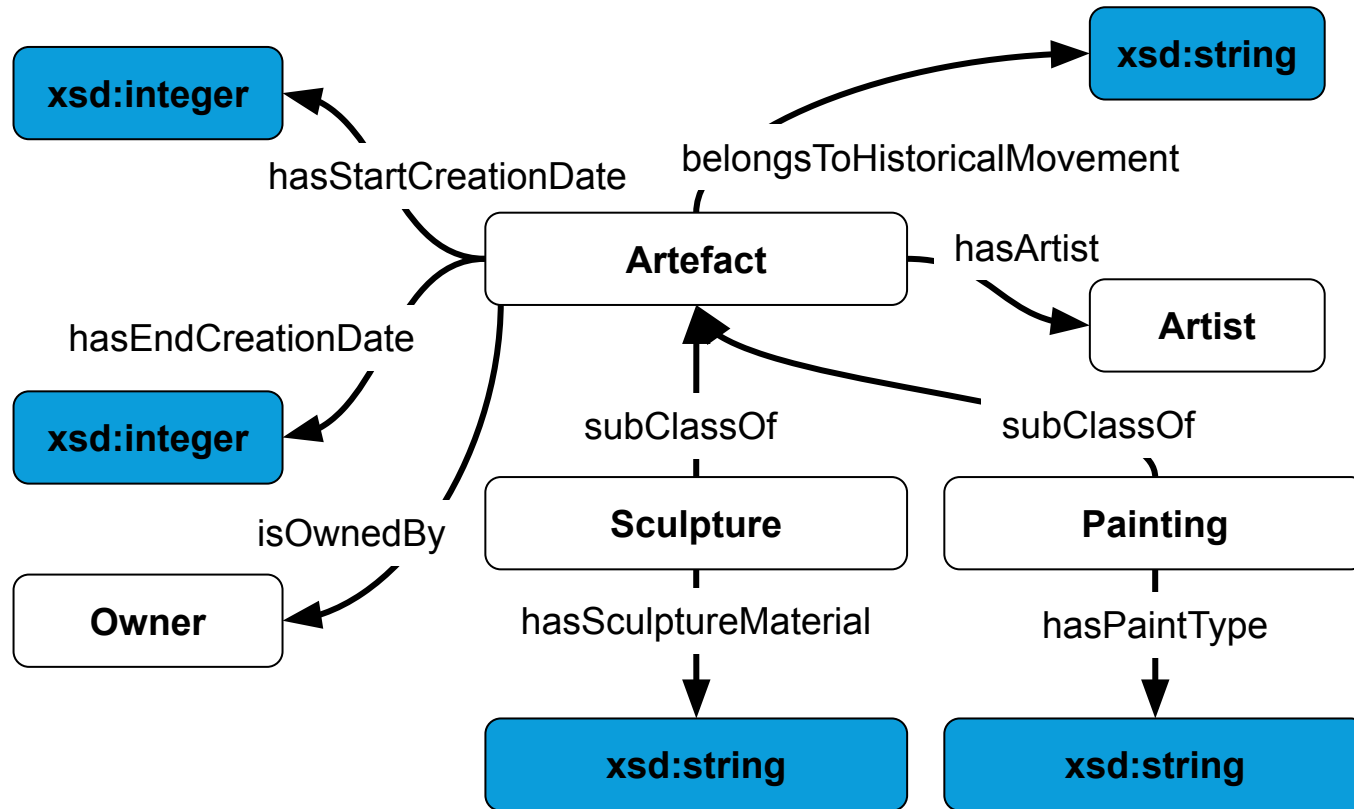
Do they need to create a new ontology? Or can they reuse?

- They can extend the ontology with owner.
- They can define a different shape.



- It allows us to define multiple shapes over the same model!
- This allows interoperability between the data of the two museums

One model, multiple shapes



Recap so far...

- **Open world:** missing information does not cause an error **OWL**
- **Closed world:** missing information causes an error **SHACL**
 - **Data Integrity : SHACL**
 - **Data Validation : SHACL**
- **Inference for Classification: OWL**
 - Class definition
 - Property domain and range
 - Subclass relationship
- **Multiple shapes / data behaviors** for the same model: **OWL + SHACL**



Let's talk about Constraints

TechAtBloomberg.com

© 2024 Bloomberg Finance L.P. All rights reserved.

Bloomberg

I want to ensure every art piece is **only** created by an artist

Can I use domain and range of ex:hasArtist?



protégé



- Open **model_09.ttl**
- Add ex:weeping_woman
- Adding wrongly hasArtist Cubism
or open **model_09_wrongArtist.ttl**
- Run the reasoner
- No error even though Cubism is wrong
- Cubism classified as an Artist! (remember inference?)

WHY? 'Violating' a domain or range constraint **does not** necessarily mean that the ontology is inconsistent or contains errors.

So what should we do ?

OWL solution: Define the two classes of Movement and Artist as disjoint

Fun Fact:

The Weeping Woman, valued at A\$1.6M in 1986. It was stolen from the National Gallery of Victoria by a group calling themselves the Australian Cultural Terrorists (ACT), who demanded funding for the arts and the creation of a \$25,000 annual art prize (The Picasso Ransom) or they would destroy the painting. The government refused to negotiate, and police were informed three weeks later that *The Weeping Woman* was stowed in a locker, unharmed.





Defining Constraints on your data with SHACL



Shape 2

```
ex:ArtefactShape a sh:NodeShape ;
  sh:targetClass ex:Artefact ;

  sh:property [
    sh:path ex:belongsToHistoricalMovement ;
    sh:datatype xsd:string ;
  ] ;

  sh:property [
    sh:path ex:hasArtist ;
    sh:class ex:Artist ;
  ] ;

  sh:property [
    sh:path ex:hasEndCreationDate ;
    sh:datatype xsd:integer ;
    sh:minCount 1 ;
    sh:maxCount 1 ;
  ] ;

  sh:property [
    sh:path ex:hasStartCreationDate ;
    sh:datatype xsd:integer ;
    sh:minCount 1 ;
    sh:maxCount 1 ;
  ] .
```

This restricts the values of ex:hasArtist to Class Artist

This ensure that every Painting has exactly 1 Start and End Creation Date

Solution: SHACL Constraints

Constraints

```
graph TD; Constraints[Constraints] --- ShapeBased[Shape Based]; Constraints --- ValueRange[Value Range]; Constraints --- Logical[Logical]; ShapeBased --- StringBased[String Based]; ShapeBased --- ValueType[Value Type]; ValueType --- Cardinality[Cardinality]; PropertyPair[Property Pair]; StringBased --- S1[sh:minLength]; StringBased --- S2[sh:maxLength]; StringBased --- S3[sh:pattern]; StringBased --- S4[sh:languageIn]; StringBased --- S5[sh:uniqueLang]; ShapeBased --- SB1[Node Shape]; ShapeBased --- SB2[Property Shape]; ValueRange --- VR1[sh:minExclusive]; ValueRange --- VR2[sh:maxExclusive]; ValueRange --- VR3[sh:minInclusive]; ValueRange --- VR4[sh:MaxExclusive]; Logical --- L1[sh:not]; Logical --- L2[sh:and]; Logical --- L3[sh:Or]; Logical --- L4[sh:Xone]; PropertyPair --- PP1[sh:equals]; PropertyPair --- PP2[sh:disjoining]; PropertyPair --- PP3[sh:lessThan]; PropertyPair --- PP4[sh:lessThanOrEauals]; Cardinality --- C1[sh:minCount]; Cardinality --- C2[sh:maxCount];
```

Shape Based

- Node Shape
- Property Shape

String Based

- sh:minLength
- sh:maxLength
- sh:pattern
- sh:languageIn
- sh:uniqueLang

Value Type

- sh:Class
- sh:datatype
- sh:nodeKind

Value Range

- sh:minExclusive
- sh:maxExclusive
- sh:minInclusive
- sh:MaxExclusive

Cardinality

- sh:minCount
- sh:maxCount

Logical

- sh:not
- sh:and
- sh:Or
- sh:Xone

Property Pair

- sh:equals
- sh:disjoining
- sh:lessThan
- sh:lessThanOrEauals

Recap so far...

- **Open world:** missing information does not cause an error **OWL**
- **Closed world:** missing information causes an error **SHACL**
 - **Data Integrity : SHACL**
 - **Data Validation : SHACL**
 - **Constraints for Validation: SHACL**
- **Inference for Classification: OWL**
 - Class definition
 - Property domain and range
 - Subclass relationship
- **Multiple shapes / data behaviors** for the same model: **SHACL**



Inference for **New (implicit) Knowledge**

TechAtBloomberg.com

© 2024 Bloomberg Finance L.P. All rights reserved.

Bloomberg

Inferring **New Knowledge**

You have the historical movement for your artefacts, does that allow you to also know which artist belongs to what movement?



- Open **model_10.ttl**
- Run the query 3.1

```
SELECT DISTINCT ?Artist ?Movement
```

```
WHERE {
```

```
  ?Artist a ex:Artist ;
```

```
  ex:belongsToHistoricalMovement ?Movement .
```

```
}
```



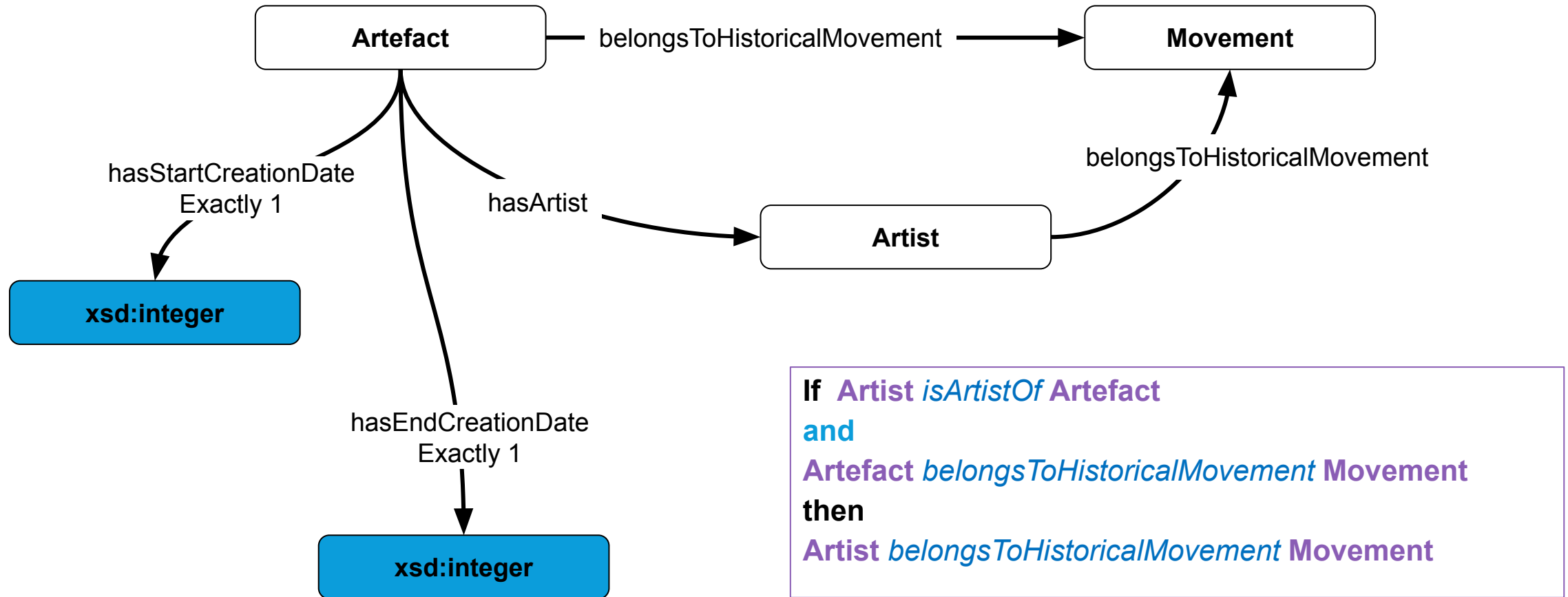
Nothing!

Solution: Property Chains!

Bloomberg

Engineering

The Model extended even more!



Property chain in OWL (owl:propertyChainAxiom)



- Convert Movement to Class
- Add *isArtistOf* inverse property
- Implement the property chain in OWL or load **model_11.ttl**
- Run the reasoner



- Save the new inferred data or upload **model_11_inferred.ttl**
- Then run the query 3.1 again



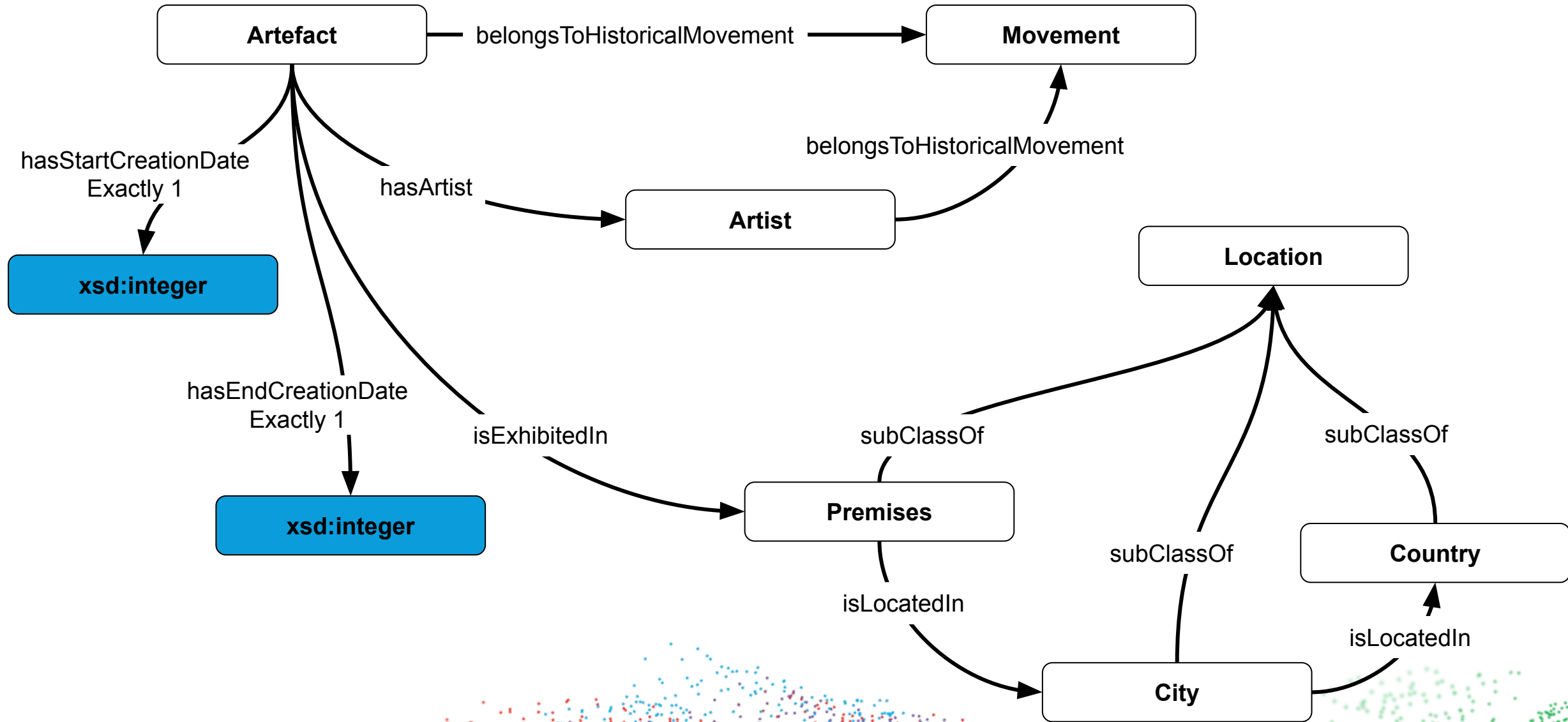
Reasoner infers the following implicit Knowledge

Property assertions: Leonardo Da Vinci	
Object property assertions +	
belongsToHistoricalMovement Renaissance	? @
isArtistOf 'Mona Lisa'	? @
isArtistOf 'The Last Supper'	? @



Artist	Movement
Leonardo Da Vinci	Renaissance
Edvard Munch	Expressionism
Pablo Picasso	Cubism
Sandro Botticelli	Renaissance
Michelangelo	Renaissance

The model extended even more!



Inferring New Knowledge

Considering that you know the museum in which the piece is exhibited, can you answer questions like:

Give me all the pieces that are located in a country (i.e., Italy)?



- Let's try the chain property again or Load **model_12.ttl**
- Run reasoner
- Save the new inferred data or upload **model_12_inferred.ttl**
- Run query 3.2



- Reasoner infers the implicit Knowledge based on property chain definition
- We are still missing the country

If **Artefact** *isExhibitedIn* **Premises**
and **Premises** *isLocatedIn* **City**
Then **Artefact** *isLocatedIn* **City**

Property assertions: David		⏏	⏏	⏏	⏏
Object property assertions +					
■ isExposedIn	'Gallery of the Academy of Florence'	?	@	x	o
■ hasArtist	Michelangelo	?	@	x	o
■ belongsToHistoricalMovement	Renaissance	?	@	x	o
■ isLocatedIn	Florence	?	@		

Transitivity in properties for inference of New Knowledge



- Let's make *isLocatedIn* transitive or Open **model_13.ttl**
- Run reasoner
- Save the new inferred data or upload **model_13_inferred.ttl**
- Run query 3.2 again



- Reasoner infers the implicit Knowledge based on property chain definition and transitivity
- You can now query the country in which the artefact is located

isLocatedIn is transitive property

Then **Artefact** *isLocatedIn* **Country**

Property assertions: David

Object property assertions +

■ isExhibitedIn 'Gallery of the Academy of Florence'



■ isLocatedIn Florence



■ hasArtist Michelangelo



■ belongsToHistoricalMovement Renaissance



■ isLocatedIn Italy



Recap so far...

- **Open world:** missing information does not cause an error **OWL**
- **Closed world:** missing information causes an error **SHACL**
 - **Data Integrity : SHACL**
 - **Data Validation : SHACL**
 - **Constraints for Validation: SHACL**
- **Inference for Classification: OWL**
 - Class definition
 - Property domain and range
 - Subclass relationship
- Inference for **new/implicit knowledge: OWL**
- **Multiple shapes / data behaviors** for the same model: **OWL+ SHACL**

TechAtBloomberg.com

© 2024 Bloomberg Finance L.P. All rights reserved.

Bloomberg

Engineering



The Material Example

TechAtBloomberg.com

© 2024 Bloomberg Finance L.P. All rights reserved.

Bloomberg

The Material Example

What are the materials used to make this artefact?

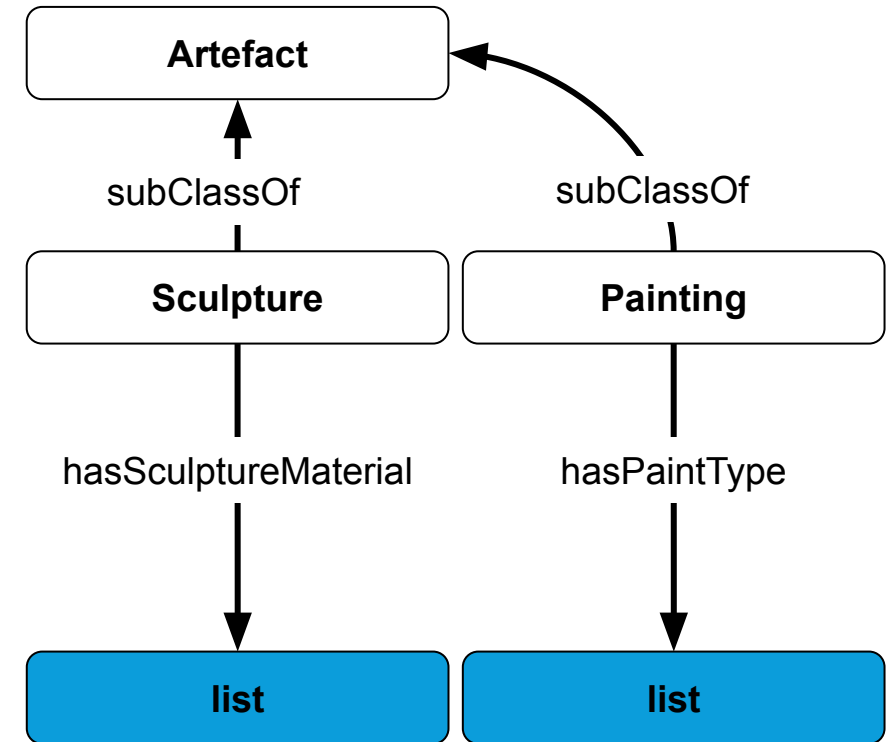
You also want to ensure that the material is correct and previously captured.

Let's start with the model

Choice 1: use **owl dataproperty** and a list as the range

Challenge:

- What if you want to say something more about each material? Like its origin or consistency?



The Material Example

What are the materials used to make this artefact?

Choice 2: Use OWL Enumeration class



- Load model_15.ttl
- Add the following wrong data
 - ex:weapingWomah ex:hasPaintType ex:florence
- Run reasoner



No Error!

Reasoner infers that florence is a member of *PaintType* class

WHY? Because of OWL's OWA



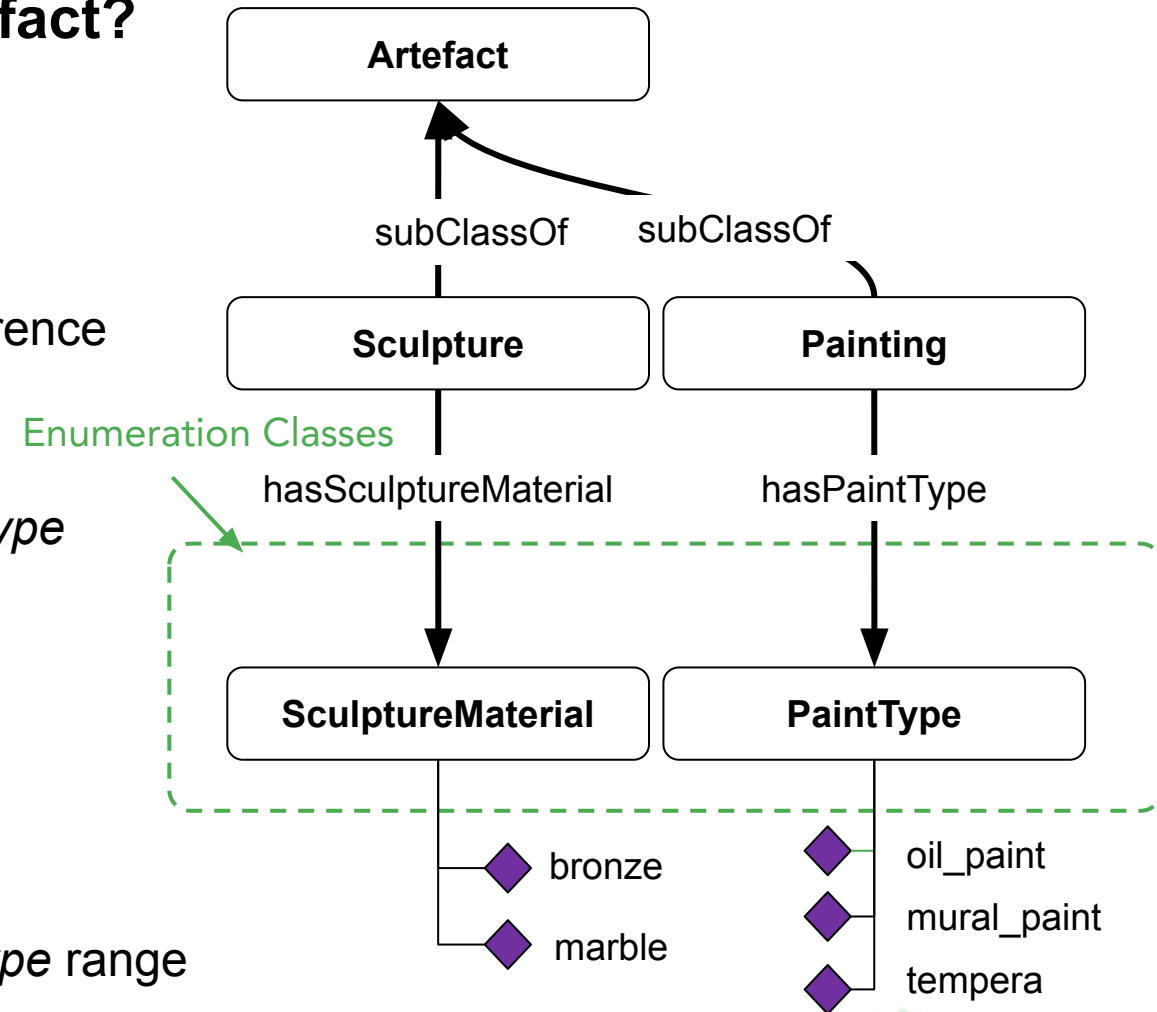
- What if we limit the range of *hasPaintType* to only one *PaintType*?



No Error!

Reasoner infers the two instances of the *hasPaintType* range are the same

WHY? OWL does not support UNA



Unique Name Assumption

- In logics with UNA, different names always refer to different entities in the world
- OWL does NOT support UNA
- Consequences:
 - Different entities have to be **declared to be different** (otherwise, they are potentially identical) - owl:differentFrom, owl:disjointWith
 - Identical entities also have to be **declared to be identical** (otherwise, they are potentially different) - owl:sameAs, owl:equivalentClass

Can we still solve this with OWL?

Yes, but...

- The property *hasPaintType* needs to be **Functional**
- The instances need to all be defined as **owl:differentFrom**

Too much work!

There's a better solution :)

The Material Example



What are the materials used to make this artefact?

Choice 3:

- Use OWL enumeration classes (model_16.ttl) & SHACL shapes to restrict the range of artefacts [Shape 10](#)

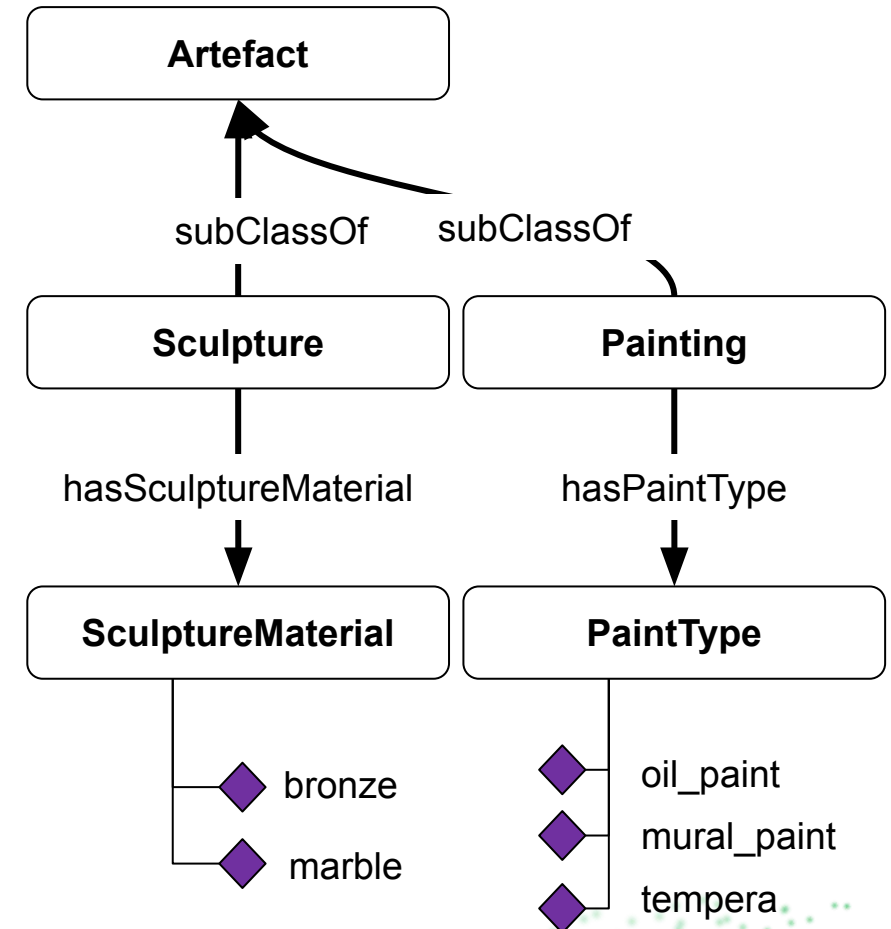
Pros: no diff, no wrong class inference, easy expansion of material types, and easy addition of information about material

```
ex:PaintingShape a sh:NodeShape ;
  sh:targetClass ex:Painting ;

  sh:property [
    sh:path ex:hasPaintType ;
    sh:class ex:PaintType ;
    sh:in (ex:oil_paint ex:mural_paint ex:tempera) ;
  ] .

ex:SculptureShape a sh:NodeShape ;
  sh:targetClass ex:Sculpture ;


  sh:property [
    sh:path ex:hasSculptureMaterial ;
    sh:class ex:SculptureMaterial ;
    sh:in (ex:marble ex:bronze) ;
  ] .
```



Full Recap

- **Open world:** missing information does not cause an error **OWL**
- **Inference for Classification:** **OWL**
 - Class definition
 - Property domain and range
 - Subclass relationship
- Inference for **new/implicit knowledge:** **OWL**
- **Closed world:** missing information causes an error **SHACL**
 - **Data Integrity:** **SHACL**
 - **Data Validation:** **SHACL**
 - Constraints for Validation
- **Multiple shapes** for the same model: **OWL + SHACL**
- **Mix of Inference and Validation:** **OWL + SHACL**





So who won ?

TechAtBloomberg.com

© 2024 Bloomberg Finance L.P. All rights reserved.

Bloomberg

WINNER :YOU



- Identify and understand the **true problem** you are trying to solve
- Understand the **strengths** and **weaknesses** of each technology and why and what it's built for
- Choose the **right tooling** and understand the strengths of your tool

Based on the above **KNOWLEDGE** infer the **RIGHT CHOICE !**

Disclaimer: Created by ChatGPT



Let's discuss more

Connect with us through LinkedIn:

- Tara: <https://www.linkedin.com/in/tara-raafat-phd-1038315/>
- Davide: <https://www.linkedin.com/in/davide~damico>

TechAtBloomberg.com

© 2024 Bloomberg Finance L.P. All rights reserved.

Bloomberg

Thank you!

Bloomberg

TechAtBloomberg.com

© 2024 Bloomberg Finance L.P. All rights reserved.